Parallel Computing

CS 1202 CIE 2 19th Feb 2013

1. Suppose that an MPI_COMM_WORLD consists of the three processes 0, 1, and 2, and suppose that the following code is executed: [5]

```
#include <stdio.h>
#include <mpi.h>

main ( int argc, char** argv ) {
  int myRank, x, y, z;
  MPI_Status status;

  MPI_Init( &argc, &argv );
  MPI_Comm_rank( MPI_COMM_WORLD, &myRank );

  switch( myRank ) {
  case 0:
    x = 0; y = 1; z = 2;
    MPI_Bcast( &x, 1, MPI_INT, 0, MPI_COMM_WORLD );
    MPI_Send( &y, 1, MPI_INT, 2, 43, MPI_COMM_WORLD );
    MPI_Bcast( &z, 1, MPI_INT, 1, MPI_COMM_WORLD );
    break;
  case 1:
    MPI_Bcast( &x, 1, MPI_INT, 0, MPI_COMM_WORLD );
    MPI_Bcast( &y, 1, MPI_INT, 1, MPI_COMM_WORLD );
    break;
  case 2:
    MPI_Bcast( &z, 1, MPI_INT, 0, MPI_COMM_WORLD );
    MPI_Recv( &x, 1, MPI_INT, 0, 43, MPI_COMM_WORLD, &status );
    MPI_Bcast( &y, 1, MPI_INT, 1, MPI_COMM_WORLD );
  }

  MPI_Finalize();
}
```

What are the values of x, y, and z on each process after the code has been executed? You must explain your answer.

2. Consider a communicator with 4 processes. How many total MPI_Send()'s and MPI_Recv()'s would be required to accomplish the following: **MPI_Allreduce (&a, &x, 1, MPI_REAL, MPI_SUM, comm); [ ]** [2]

a) 3
b) 4
c) 12
d) 16

3. Three processors have the following data in an array in each processor: [2]

Proc0	*Proc1*	*Proc2*
a(1) = 1	a(1) = 2	a(1) = 4
	a(2) = 3	a(2) = 5
		a(3) = 6

After a specific MPI routine call the data gets distributed as follows:

Proc0	*Proc1*	*Proc2*
b(1) = 1	b(1) = 1	b(1) = 1
b(2) = 2	b(2) = 2	b(2) = 2
b(3) = 3	b(3) = 3	b(3) = 3
b(4) = 4	b(4) = 4	b(4) = 4
b(5) = 5	b(5) = 5	b(5) = 5
b(6) = 6	b(6) = 6	b(6) = 6

What is this MPI routine called? Explain the algorithm of the routine along with the complexity in brief.

4. Performance analysis: [4]
The execution time of Floyd's algorithm is computed in the textbook as:

$$n^2\left\lceil\frac{n}{p}\right\rceil\chi + n(\log p)\lambda + \lceil\log p\rceil 4\frac{n}{\beta}$$

Here, we assume that n=1000, χ = 25.5 nsec, λ= 250 μsec, and β = 10 7.
The value of this formula for p=1, 2, 3, 4, 5, 6, 7, and 8 is respectively:
25.5000, 13.0000, 9.0170, 6.8750, 5.8500, 5.0085, 4.3965, and 3.9375
Compute the Karp-Flatt serial fraction for p=2, 4, and 8. Explain the meaning of the values obtained for the serial fraction. Hint: Watch the units!

OR

---

5. Discuss in brief the computation and communication structures involved in the optimized parallel Sieve of Erathosthenes algorithm along with their respective complexities. The main computation in this algorithm is done in a two level loop, why would interchanging the loops improves performance ? [4]

5. You are being the scientific computing advisor at a supercomputer centre, you are faced with a physics researcher looking for suggestions from you on efficient parallelization of a highly compute intensive physics code. After profiling you observed that matrix vector multiplication involving large matrices is the major time consuming computation in the code. The researchers says he has two ready to use ways of implementing the MVM code, i). Rowwise block-striped decomposition of the matrix and with replicating a block-mapped vector and ii). Columnwise block-striped decomposition of the matrix and block decomposed vector. After describing the computation and communication requirements of both the algorithms very briefly to the customer, which one would you recommend him to use in his code ? [7]

6. What is the output of the following program? Express/explain your answer in terms of P, the number of processes on which the program executes. Assume P is a power of two. Explain the behavior of the program in one or two sentences. [5]

```
#include <cstdio>
#include "mpi.h"

int main(int argc, char* argv[])
{
  int rank, received, myvalue, p, neighbor;
  MPI_Init(&argc,&argv);
  MPI_Status status;
  MPI_Comm_size(MPI_COMM_WORLD,&p);
  MPI_Comm_rank(MPI_COMM_WORLD,&rank);
  myvalue = rank;
  for (int step = 1; step < p; step = step * 2) {
    if (rank % step == 0) {              // % is the modulo operator
      if (isEven(rank / step)) {
         // isEven(x) returns true (or 1) if x is even,
         // and false (or 0) otherwise.
        MPI_Recv(&received,1,MPI_INT,rank+step,0,MPI_COMM_WORLD,&status);
        myvalue += received;
      } else {
        MPI_Send(&myvalue,1,MPI_INT,rank-step,0,MPI_COMM_WORLD);
      }
    } else {
      break; // jump out of the for loop
    }
  }
  if (rank == 0) {
    printf("%d\n",myvalue); // print myvalue to the screen
  }
  return 0;
}
```

OR

---

7. What is the algorithm used and the complexity of one-to-all broadcast ? Write about an improved version of one-to-all broadcast? Justify that the improved version is better than the original ? [5]